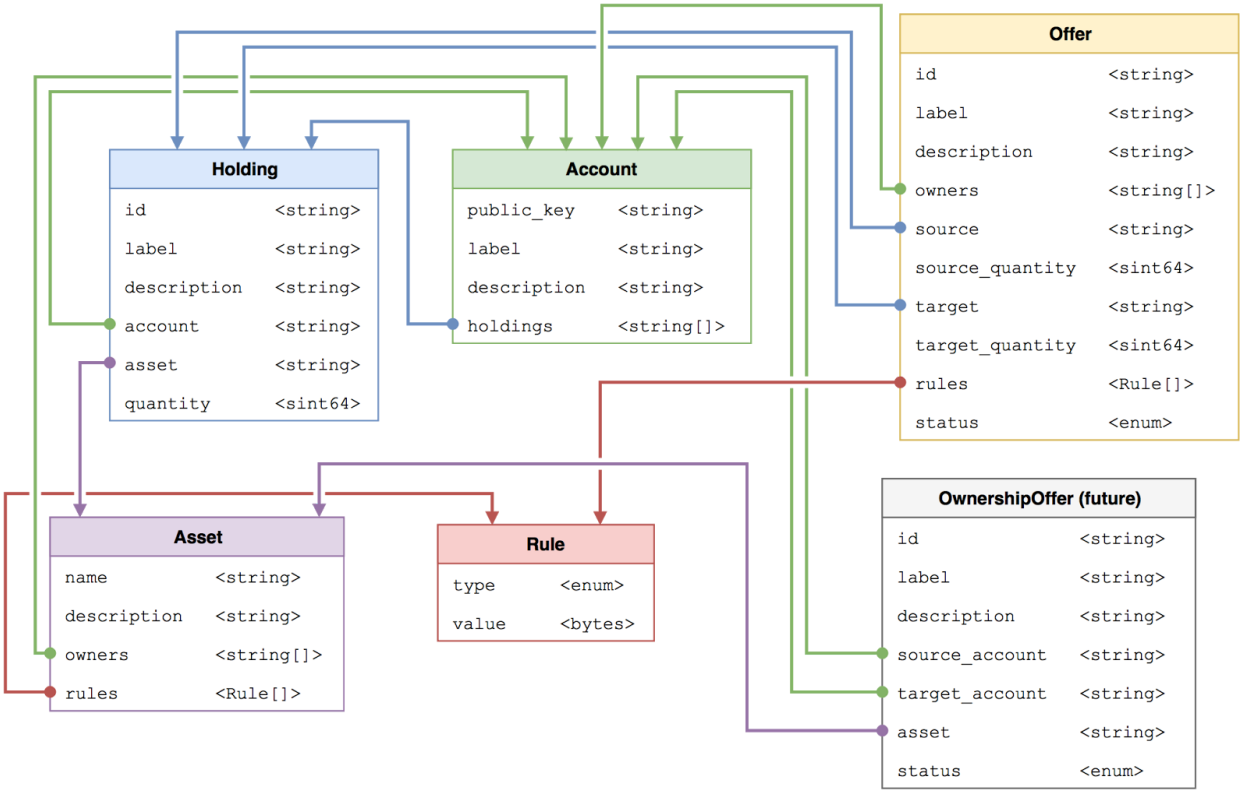
Marketplace 2.0 Design

Zac Delventhal
created 11-17-2017
updated 12-26-2017

Contents

State Components



State Protos

These Protobuf messages represent the actual objects stored in blockchain state. They match the diagram above, with the addition of containers which will be used to handle hash collisions in addressing.

```
message Account {
    string public_key = 1;
    string label = 2;
    string description = 3;
    repeated string holdings = 4;
}

message AccountContainer {
    repeated Account entries = 1;
}

message Holding {
```

```
    string id = 1;
    string label = 2;
    string description = 3;
    string account = 4;
    string asset = 5;
    sint64 quantity = 6;
}

message HoldingContainer {
    repeated Holding entries = 1;
}

message Asset {
    string name = 1;
    string description = 2;
    repeated string owners = 3;
    repeated Rule rules = 4;
}

message AssetContainer {
    repeated Asset entries = 1;
}

message Offer {
    enum Status {
        STATUS_UNSET = 0;
        OPEN = 1;
        CLOSED = 2;
    }

    string id = 1;
    string label = 2;
    string description = 3;
    repeated string owners = 4;
    string source = 5;
    sint64 source_quantity = 6;
    string target = 7;
    sint64 target_quantity = 8;
    repeated Rule rules = 9;
    Status status = 10;
}

message OfferContainer {
    repeated Offer entries = 1;
}

message OwnershipOffer {
```

```
    enum Status {
       STATUS_UNSET = 0;
       OPEN = 1;
       CLOSED = 2;
    }

    string id = 1;
    string label = 2;
    string description = 3;
    sint64 source_account = 4;
    sint64 target_account = 5;
    string asset = 6;
    Status status = 7;
}

message OwnershipOfferContainer {
    repeated OwnershipOffer entries = 1;
}
```

**OwnershipOffers is a likely future addition to allow expanding asset ownership beyond the initial creator. However, this feature will not be included in the initial release.*

Rule Proto

The Rule protobuf deserves special attention. It is designed to be an extensible way to add rules and special conditions to both Assets and Offers. The protobuf has two parts, a type enum which describes its function, and a bytes value, which might be omitted, could be a boolean or an integer, or could potentially contain executable smart contract code.

Some Notes:

- Rules from 100-199 are Asset rules, and Rules from 200-299 are Offer rules
- An Asset can have both types of rules, and any Offer rules they have will automatically be added to Offers created with that Asset
- Offers can only have Offer rules
- "Hooks" are designations for when a bit of smart code should be inserted into the validation process. This is just an early idea, and will not be included in the initial release.
- Other greyed out rules are also intended for future releases.

```
message Rule {
    enum RuleType {
        RULE_UNSET = 0;

        OWNER_HOLDINGS_INFINITE = 100;
        ALL_HOLDINGS_INFINITE = 101;
        NOT_TRANSFERABLE = 102;
```

```
        EXCHANGE_FOR_ASSETS = 103;
        EXCHANGE_WITH_RATIOS = 104;
        PRE_HOLDING_HOOK = 150;
        POST_HOLDING_HOOK = 151;
        PRE_OFFER_HOOK = 152;
        POST_OFFER_HOOK = 153;

        EXCHANGE_ONCE = 200;
        EXCHANGE_ONCE_PER_ACCOUNT = 201;
        EXCHANGE_LIMITED_TO_ACCOUNTS = 202;
        EXCHANGE_N_TIMES = 203;
        EXCHANGE_N_TIMES_PER_ACCOUNT = 204;
        PRE_EXCHANGE_HOOK = 250;
        POST_EXCHANGE_HOOK = 251;
    }

    RuleType type = 1;
    bytes value = 2;
}
```

Asset Rules

Type: *OWNER_HOLDINGS_INFINITE*
Value: *None*
For "owners" of this Asset, the "quantity" property in their Holdings should be considered as essentially a boolean. Either they have a quantity of that Asset, or they do not. Operations which would modify the quantity should be ignored *unless* it is to go from 0 to a different quantity.

Type: *ALL_HOLDINGS_INFINITE*
Value: *None*
For all Holdings of this Asset, the "quantity" property should be considered as essentially a boolean. Either there is a quantity of that Asset, or there is not. Operations which would modify the quantity should be ignored *unless* it is to go from 0 to a different quantity.

Type: *EXCHANGE_FOR_ASSETS*
Value: *List of Asset names*
Offers created with this Asset are only valid if it is being exchanged for one of the assets listed.

Type: *EXCHANGE_WITH_RATIOS*
Value: *List of integer tuples*
Offers created with this Asset must have source/target quantities which match one of the listed ratios, where the first integer is the quantity of this Asset, and the second is the quantity of the other Asset. If used in conjunction with *EXCHANGE_FOR_ASSETS*, and the index of a ratio corresponds with the index of an Asset, then Offers will be limited to an exchange of *that* Asset at the specified ratio.

Type: *PRE_HOLDING_HOOK*
Value: *Bytecode*
Smart contract code which will fire before creating a Holding with this Asset.

Type: *POST_HOLDING_HOOK*
Value: *Bytecode*
Smart contract code which will fire after creating a Holding with this Asset.

Type: *PRE_OFFER_HOOK*
Value: *Bytecode*
Smart contract code which will fire before creating an Offer with this Asset.

Type: *POST_OFFER_HOOK*
Value: *Bytecode*
Smart contract code which will fire after creating an Offer with this Asset.

Offer Rules

Type: *EXCHANGE_ONCE*
Value: *None*
This Offer may only be accepted once before it is automatically closed.

Type: *EXCHANGE_ONCE_PER_ACCOUNT*
Value: *None*
Each Account may accept this Offer once. Repeated attempts to accept the Offer by the same Account will be invalid.

Type: *EXCHANGE_LIMITED_TO_ACCOUNTS*
Value: *List of Account keys*
This Offer may only be accepted by the Accounts listed. Attempts to accept the Offer by other Accounts will be invalid.

Type: *EXCHANGE_N_TIMES*
Value: *Integer*
This Offer will automatically be closed after being accept the specified number of times.

Type: *EXCHANGE_N_TIMES_PER_ACCOUNT*
Value: *Integer*
Each Account may accept this Offer up to the specified number of times. Attempts to accept the Offer by Accounts which have hit this limit will be invalid.

Type: *PRE_EXCHANGE_HOOK*
Value: *Bytecode*
Smart contract code which will fire before each time an Offer is accepted.

Type: *POST_EXCHANGE_HOOK*
Value: *Bytecode*
Smart contract code which will fire after each time an Offer is accepted.

Payload Protos

All Transaction payloads will be serialized *MarketplacePayload* Protobuf messages. The processor uses a *PayloadType* enum to determine which property contains the logic to execute. Anyone creating transactions must be sure that the enum and property they set match. As with previous sections, all greyed out payload types are intended for future versions of Marketplace, and will not initially be implemented.

```
message TransactionPayload {

    enum PayloadType {
        TYPE_UNSET = 0;

        CREATE_ACCOUNT = 2;
        CREATE_ASSET = 3;
        CREATE_HOLDING = 4;
        CREATE_OFFER = 5;
        CREATE_OWNERSHIP_OFFER = 6;

        ACCEPT_OFFER = 10;
        CLOSE_OFFER = 11;
        ACCEPT_OWNERSHIP_OFFER = 12;
        CLOSE_OWNERSHIP_OFFER = 13;
        UPDATE_ACCOUNT = 14;
        UPDATE_HOLDING = 15;
        REMOVE_OWNERSHIP = 16;
    }

    PayloadType payload_type = 1;

    CreateAccount create_account = 2;
    CreateAsset create_asset = 3;
    CreateHolding create_holding = 4;
    CreateOffer create_offer = 5;
    CreateOwnershipOffer create_ownership_offer = 6;

    AcceptOffer accept_offer = 10;
    CloseOffer close_offer = 11;
    AcceptOwnershipOffer accept_ownership_offer = 12;
    CloseOwnershipOffer close_ownership_offer = 13;
    UpdateAccount update_account = 14;
```

```
    UpdateHolding update_holding = 15;
    RemoveOwnership remove_ownership = 16;
}

message CreateAccount {
    string label = 1;
    string description = 2;
}

message CreateAsset {
    string name = 1;
    string description = 2;
    repeated Rule rules = 3;
}

message CreateHolding {
    string id = 1;
    string label = 2;
    string description = 3;
    string asset = 4;
    sint64 quantity = 5;
}

message CreateOffer {
    string id = 1;
    string label = 2;
    string description = 3;
    string source = 4;
    sint64 source_quantity = 5;
    string target = 6;
    sint64 target_quantity = 7;
    repeated Rule rules = 8;
}

message CreateOwnershipOffer {
    string id = 1;
    string label = 2;
    string description = 3;
    string target_account = 4;
    string asset = 5;
}

message AcceptOffer {
    string id = 1;
    string source = 2;
    string target = 3;
    uint64 count = 4;
```

```
}

message CloseOffer {
    string id = 1;
}

message AcceptOwnershipOffer {
    string id = 1;
}

message CloseOwnershipOffer {
    string id = 1;
}

message UpdateAccount {
    string label = 1;
    string description = 2;
}

message UpdateHolding {
    string label = 1;
    string description = 2;
}

message RemoveOwnership {
    string asset = 1;
}
```